Една от често използваните формули за изчисляване на дневните калорийни нужди е Харрис-Бенедикт. Тя взема предвид пол, възраст, ръст и тегло. За мъже: BMR = 88.362 + (13.397 x тегло в кг) + (4.799 x височина в см) – (5.677 x възраст в години). За жени: BMR = 447.593 + (9.247 x тегло в кг) + (3.098 x височина в см) – (4.330 x възраст в години). Полученият BMR (базов метаболитен ритъм) се умножава по фактор на активност за да се определи общия дневен калориен прием.

Шизгал актуализират формулата на Харис-Бенедикт на базата на нови данни, като променят заложените в нея коефициенти: Мъже: BMR = 88,362 + (13,397 x тегло в кг) + (4,799 x ръст в см) – (5,677 x възраст в години) Жени: BMR = 447,593 + (9,247 x тегло в кг) + (3,098 x ръст в см) - (4,330 x възраст в години)

**BMR** (базов метаболитен ритъм) е количеството калории, което тялото ви изразходва в покой, за да поддържа основните функции на организма, като дишане, кръвообращение и работата на органите. Той е важен показател, който помага при изчисляването на калорийния баланс за сваляне или покачване на тегло.

**Как се изчислява BMR?**

Съществуват различни формули за изчисляване на BMR, но най-често използваните са формулите на **Харис-Бенедикт** и **Мифлин-Сент Жор**.

**1. Формула на Харис-Бенедикт**

* **За мъже**:

BMR = 88.362 + (13.397×тегло в кг) + (4.799×височина в см) − (5.677×възраст в години)

* **За жени**:

BMR = 447.593 + (9.247×тегло в кг) + (3.098×височина в см) − (4.330×възраст в години)

**2. Формула на Мифлин-Сент Жор**

* **За мъже**:

BMR=(10×тегло в кг)+(6.25×височина в см)−(5×възраст в години)+5

**За жени**:

BMR=(10×тегло в кг)+(6.25×височина в см)−(5×възраст в години)−161

**Как да определите общите калории за деня?**

След като изчислите BMR, можете да умножите резултата по фактор на активност, за да определите общите калорийни нужди:

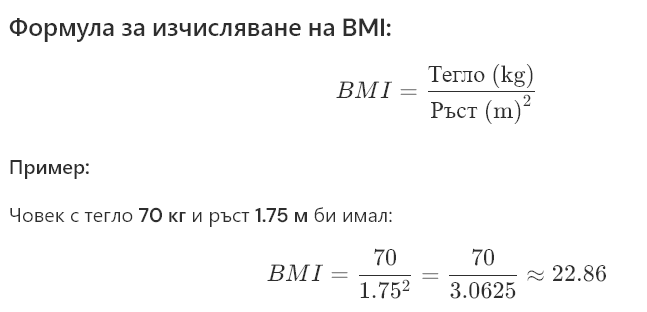
* **Седящ начин на живот** (малко или никакво упражнение): BMR × 1.2
* **Леко активен** (леко упражнение 1-3 дни в седмицата): BMR × 1.375
* **Умерено активен** (умерени упражнения 3-5 дни в седмицата): BMR × 1.55
* **Много активен** (тежки упражнения 6-7 дни в седмицата): BMR × 1.725
* **Екстремно активен** (твърд физически труд или тренировки два пъти на ден): BMR × 1.9
* когато секцията се скрива чрез v-if в Vue.js, DOM елементите вътре в секцията (включително слайдера) реално се премахват от DOM. Когато секцията отново стане активна (active\_section == 0) и DOM елементите се рендерират наново, Owl Carousel библиотеката не знае, че трябва да инициализира слайдера отново.
* За да решите този проблем, трябва да се инициализира слайдерът повторно, след като секцията стане видима отново. Може да използвате Vue lifecycle hooks или watcher, за да засечете промяната в active\_section и да инициализирате слайдера отново.

Ключовите стъпки са следните:

1. Преинициализирайте слайдера с this.initSlider(), когато active\_section == 0 (чрез watch).
2. Унищожете слайдера при скриване на секцията с this.destroySlider().
3. Използвайте this.$nextTick(), за да гарантирате правилно DOM рендериране преди инициализация.
4. **setBackgroundImages**: Тази функция се грижи фоновите изображения да бъдат зададени правилно, използвайки атрибута data-setbg.
5. **initSlider**: Тази функция комбинира настройката на фоновите изображения и инициализацията на Owl Carousel.
6. **destroySlider**: Опционално премахва слайдера, когато секцията вече не е активна, за да избегнете нежелани конфликти.

**BMI** означава **Body Mass Index** (индекс на телесната маса), който е широко използван инструмент за определяне на здравословното телесно тегло във връзка с ръста на човека. Това е базова мярка, която не взима под внимание количеството мазнини, мускулна маса или специфични характеристики на тялото, но може да предостави обща идея за теглото на човек спрямо неговия ръст.

### Формула за изчисляване на BMI:



### Категории според BMI:

Съгласно **Световната здравна организация (WHO)**, стойностите на BMI се категоризират така:

| **Категория** | **BMI** |
| --- | --- |
| Недостатъчно тегло | Под 18.5 |
| Нормално тегло | 18.5 – 24.9 |
| Наднормено тегло | 25.0 – 29.9 |
| Затлъстяване степен I | 30.0 – 34.9 |
| Затлъстяване степен II | 35.0 – 39.9 |
| Затлъстяване степен III | Над 40.0 |

### Ограничения на BMI:

Въпреки че е добър ориентир, индексът на телесната маса не винаги показва точното здравословно състояние, защото:

* **Не отчита мускулната маса:** Мускулестите хора (например спортисти) може да имат висок BMI, но ниски нива на мазнини.
* **Не отчита разпределението на мазнините:** BMI не различава мазнините около талията (които са свързани със здравословни рискове) от тези на други места.
* **Пренебрегва индивидуалните параметри:** Например възраст, пол, етническа принадлежност и състояние на организма (като бременност).

### Кога се използва BMI?

BMI често се използва:

1. **В здравеопазването:** За първична оценка на риска от заболявания, свързани с теглото (като диабет, сърдечно-съдови заболявания и др.).
2. **Във фитнес програми:** За да се проследява прогреса на клиентите.
3. **В хранителни режими:** За да се определи нуждата от качване или сваляне на тегло.

### Как да изчислите BMI програмно?

Ето примерен код на JavaScript за изчисляване на BMI:

function calculateBMI(weight, height) {

const bmi = weight / (height \*\* 2); // Тегло (kg) / Ръст (m)^2

return bmi.toFixed(2); // Закръгляне до две цифри след десетичната точка

}

// Пример

const weight = 70; // kg

const height = 1.75; // m

console.log("Вашият BMI е: " + calculateBMI(weight, height));

в кода на HTML страницата въвеждам възраст: <input id="Age" type="text" placeholder="Възраст" v-model="age">. Използвам vue.js и age приема stringе като стойност. Искам да го преобразувам в число, но с отчитане на възмпжна грешка при преобразуването и присвояване в този случай на стойност по подразбиране (например 0)

**GPT-4o**

Във Vue.js, когато искаме да преобразуваме стойност от текстовото поле в число и същевременно да обработим потенциални грешки (напр. невалидна стойност), можем да решим проблема чрез **методи** или да използваме **computed property**. Ето как да го реализирате:

### Решение 1: Използване на ****computed property****

Можете да създадете изчислимо свойство, което обработва стойността на age (превръща я в число) и задава стойност по подразбиране (например 0), ако преобразуването не е успешно.

<input id="Age" type="text" placeholder="Възраст" v-model="age">

<p>Преобразувана възраст: {{ numericAge }}</p>

export default {

data() {

return {

age: "" // Стойността ще бъде текст по подразбиране (string)

};

},

computed: {

numericAge() {

const parsedAge = parseInt(this.age, 10);

return isNaN(parsedAge) ? 0 : parsedAge; // Ако не може да се преобразува, връща 0

}

}

};

#### Обяснение:

1. parseInt(this.age, 10): Превръща въведения текст от полето в число (цели числа, база 10).
2. isNaN(parsedAge): Проверява дали преобразуването е неуспешно (ако стойността не е число).
3. Ако не е валидно число, връща стойност по подразбиране (**0** тук).

### Решение 2: Използване на watcher

Ако искате директно да преобразувате и зададете числова стойност на age (с промяна на оригиналното свойство age), можете да използвате **watcher**.

<input id="Age" type="text" placeholder="Възраст" v-model="age">

<p>Стойност за възраст: {{ age }}</p>

export default {

data() {

return {

age: "" // Текст по подразбиране

};

},

watch: {

age(newValue) {

// Преобразуване на стойността в число

const parsedAge = parseInt(newValue, 10);

// Проверка за валидност. Ако е невалидно, задаваме стойност по подразбиране (0)

this.age = isNaN(parsedAge) ? 0 : parsedAge;

}

}

};

#### Обяснение:

1. Watcher-ът засича промените в age всеки път, когато потребителят въвежда нещо в текстовото поле.
2. Преобразуваме въведената стойност с parseInt.
3. Ако получената стойност не е валидно число (чрез isNaN), задаваме стойността директно на **0**.
4. Тази корекция автоматично се обновява в полето за въвеждане.

### Решение 3: Директно в събитието (например, при загуба на фокус)

Можете да използвате събитие @blur (или например @input), за да обработите преобразуването само когато потребителят приключи с въвеждането.

<input id="Age" type="text" placeholder="Възраст" v-model="age" @blur="sanitizeAge">

<p>Стойност за възраст: {{ age }}</p>

export default {

data() {

return {

age: "" // Текст по подразбиране

};

},

methods: {

sanitizeAge() {

const parsedAge = parseInt(this.age, 10);

this.age = isNaN(parsedAge) ? 0 : parsedAge; // Ако е невалидно, задаваме 0

}

}

};

#### Обяснение:

1. @blur="sanitizeAge": Събитието blur се задейства, когато потребителят премести курсора извън полето за въвеждане.
2. Методът sanitizeAge преобразува age в число или задава стойност по подразбиране (0) при грешка.

### Защо да изберете определено решение?

1. **Computed property:** Препоръчително, ако не искате да модифицирате директно стойността на age, а само да я използвате в числов вид другаде (например за изчисления).
2. **Watcher:** Автоматично преобразува и обновява стойността на age. Подходящо, ако работите с изчистени числови стойности.
3. **Събитие @blur:** Добър избор, когато искате преобразуването да стане само при загуба на фокус (а не при всяка промяна).

Ако имате други въпроси или специфични изисквания, можете да споделите!